# Widening for MDL-based Retail Signature Discovery

Clément Gautrais[1][0000−0001−8486−9616]*, Peggy Cellier[2], Matthijs van Leeuwen[3], and Alexandre Termier[2]

[1] KU Leuven, Department of Computer Science, Leuven, Belgium
clement.gautrais@cs.kuleuven.be
[2] Univ Rennes, Inria, INSA, CNRS, IRISA
[3] LIACS, Leiden University, Leiden, the Netherlands

**Abstract.** *Signature patterns* have been introduced to model repetitive behavior, e.g., of customers repeatedly buying the same set of products in consecutive time periods. A disadvantage of existing approaches to signature discovery, however, is that the required number of occurrences of a signature needs to be manually chosen. To address this limitation, we formalize the problem of selecting the best signature using the minimum description length (MDL) principle. To this end, we propose an encoding for signature models and for any data stream given such a signature model. As finding the MDL-optimal solution is unfeasible, we propose a novel algorithm that is an instance of *widening*, i.e., a diversified beam search that heuristically explores promising parts of the search space. Finally, we demonstrate the effectiveness of the problem formalization and the algorithm on a real-world retail dataset, and show that our approach yields relevant signatures.

**Keywords:** signature discovery · minimum description length · widening

## 1 Introduction

When analyzing (human) activity logs, it is especially important to discover recurrent behavior. Recurrent behavior can indicate, for example, personal preferences or habits, and can be useful in contexts such as personalized marketing. Some types of behavior are elusive to traditional data mining methods: for example, behavior that has some temporal regularity but not strong enough to be periodic, and which does not form simple itemsets or sequences in the log. A prime example is the set of products that is essential to a retail customer: all of these products are bought regularly, but often not periodically due to different depletion rates, and they are typically bought over several transactions—in any arbitrary order—rather than all at the same time.

To model and detect such behavior, we have proposed *signature patterns* [3]: patterns that identify irregular recurrences in an event sequence by segmenting the sequence (see Figure 1). We have shown the relevance of signature patterns

* This work has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models)

in the retail context, and demonstrated that they are general enough to be used in other domains, such as political speeches [2]. As a disadvantage, however, signature patterns require the analyst to provide the number of recurrences, i.e., the number of segments in the segmentation. This number of segments influences the signature: fewer segments give a more detailed signature, while more segments result in a simpler signature. Although in some cases domain experts may have some intuition on how to choose the number of segments, it is often difficult to decide on a good trade-off between the number of segments and the complexity of the signature. The main problem that we study in this paper is therefore how to automatically set this parameter in a principled way, based on the data.

Our first main contribution is a problem formalization that defines the best signature for a given dataset, so that the analyst no longer needs to choose the number of segments. By considering the signature corresponding to each possible number of segments as a model, we can naturally formulate the problem of selecting the best signature as a model selection problem. We formalize this problem using the minimum description length (MDL) principle [4], which, informally, states that the best model is the one that compresses the data best. The MDL principle perfectly fits our purposes because 1) it allows to select the simplest model that adequately explains the data, and 2) it has been previously shown to be very effective for the selection of pattern-based models (e.g., [11, 7]).

After defining the problem using the MDL principle, the remaining question is how to solve it. As the search space of signatures is extremely large and the MDL-based problem formulation does not offer any properties that could be used to substantially prune the search space, we resort to heuristic search. Also here, the properties of signature patterns lead to technical challenges. In particular, we empirically show that a naïve beam search often gets stuck in suboptimal solutions. Our second main contribution is therefore to propose a diverse beam search algorithm, i.e., an instance of *widening* [9], that ensures that a diverse set of candidate solutions is maintained on each level of the beam search. For this, we define a distance measure for signatures based on their segmentations.

## 2 Preliminaries

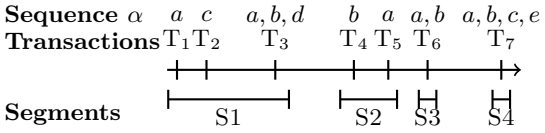


**Fig. 1.** A sequence of transactions and a 4-segmentation. We have the signature items $\mathcal{R} = \{a, b\}$, the remaining items $\mathcal{E} = \{c, d, e\}$, the set of items $\mathcal{I} = \{a, b, c, d, e\}$, the segmentation $S = \langle [T_1, T_2, T_3], [T_4, T_5], [T_6], [T_7] \rangle$.

*Signatures.* Let us first recall the definition of a *signature* as presented in [3]. Let $\mathcal{I}$ be the set of all items, and let $\alpha = \langle T_1 \dots T_n \rangle$, $T_i \subseteq \mathcal{I}$ be a sequence of itemsets. A *k-segmentation* of $\alpha$, denoted $S(\alpha, k) = \langle S_1 \dots S_k \rangle$, is a sequence of $k$ non-overlapping consecutive sub-sequences of $\alpha$, denoted $S_i$ and called *segments*, each consisting of consecutive transactions. An example of a 4-segmentation is given in Figure 1. Given $S(\alpha, k) = \langle S_1 \dots S_k \rangle$, a $k$-segmentation of $\alpha$, we have $Rec(S(\alpha, k)) = \bigcap_{S_i \in S(\alpha,k)} (\bigcup_{T_j \in S_i} T_j)$: the set of all recurrent items that are present in each segment of $S(\alpha, k)$. For example in Figure 1, the segmentation $S(\alpha, 4) = \langle S_1, S_2, S_3, S_4 \rangle$ gives $Rec(S(\alpha, 4)) = \{a, b\}$. Given $k$ and $\alpha$, one can compute $S_{max}(\alpha, k)$, the set of $k$-segmentation of $\alpha$ yielding the largest sets of recurrent items: $S_{max}(\alpha, k) = \text{argmax}_{S(\alpha,k)} |Rec(S(\alpha, k))|$. For example, in Figure 4, $\langle S_1, S_2, S_3, S_4 \rangle$ is the only 4-segmentation yielding two recurrent items. As all other 4-segmentations either yield zero or one recurrent item, $S_{max}(\alpha, 4) = \{\langle S_1, S_2, S_3, S_4 \rangle\}$. A $k$-signature (also named signature when $k$ is clear from context) is then defined as a maximal set of recurrent items in a $k$-segmentation $S$, with $S \in S_{max}(\alpha, k)$. As $S_{max}(\alpha, k)$ can contain several segmentations, we define the $k$-signature set $Sig(\alpha, k)$, which contains all $k$-signatures: $Sig(\alpha, k) = \{Rec(S_m(\alpha, k)) \mid S_m \in S_{max}(\alpha, k)\}$. $k$ gives the number of recurrences of the recurrent items in sequence $\alpha$. Given a number of recurrences $k$, finding a *k-signature* relies on finding a $k$-segmentation that maximizes the size of the itemset that occurs in each segment of that segmentation. For example, in Figure 1, given segmentation $S = \langle S_1, S_2, S_3, S_4 \rangle$ and given that $S_{max}(\alpha, 4) = \{S\}$, we have $Sig(\alpha, 4) = \{Rec(S)\} = \{\{a, b\}\}$. For simplicity, the segmentation associated with a $k$-signature in $Sig(\alpha, k)$ is denoted $S = \langle S_1 \dots S_k \rangle$, and the signature items are denoted $\mathcal{R} \subseteq \mathcal{I}$. The remaining items are denoted $\mathcal{E}$, i.e., $\mathcal{E} = \mathcal{I} \setminus \mathcal{R}$.

*Minimum description length (MDL).* Let us now briefly introduce the basic notions of the minimum description length (MDL) principle [4] as it is commonly used in compression-based pattern mining [7]. Given a set of models $\mathcal{M}$ and a dataset $\mathcal{D}$, the best model $M \in \mathcal{M}$ is the one that minimizes $L(\mathcal{D}, M) = L(M) + L(\mathcal{D}|M)$, with $L(M)$ the length, in bits, of the encoding of $M$, and $L(\mathcal{D}|M)$ the length, in bits, of the encoding of the data given $M$. This is called *two-part MDL* because it separately encodes the model and the data given the model, which results in a natural trade-off between model complexity and data complexity. To fairly compare all models, the encoding has to be *lossless*. To use the MDL principle for model selection, the model class $\mathcal{M}$ has to be defined (in our case, the set of all signatures), as well as how to compute the length of the model and the length of the data given the model. It should be noted that only the *encoded length* of the data is of interest, not the encoded data itself.

## 3 Problem Definition

To extract recurrent items from a sequence using signatures, one must define the number of segments $k$. Providing meaningful values for $k$ usually requires expert knowledge and/or many tryouts, as there is no general rule to automatically set

$k$. Our problem is therefore to devise a method that adjusts $k$, depending on the data at hand. As this is a typical model selection problem, our approach relies on the minimum description length principle (MDL) to find the best model from a set of candidate models. However, the signature model must be refined into a probabilistic model to use the MDL principle for model selection. Especially, the occurrences of items in $\alpha$ should be defined according to a probability distribution. With no information about these occurrences, the uniform distribution is the most natural choice. Indeed, without information on the transaction in which an item occurs, the best is to assume it can occur uniformly at random in any transaction of the sequence $\alpha$. Moreover, the choice of the uniform distribution has been shown to minimize the worst case description length [4].

To make the signature model probabilistic, we assume that it generates three different types of occurrences independently and uniformly. As the signature gives the information that there is at least one occurrence of every signature item in every segment, the first type of occurrences correspond to this one occurrence of signature items in every segment. These are generated uniformly over all the transactions of every segment. The second type of occurrences are the remaining signature items occurrences. Here, the information is that these items already have occurrences generated by the previous type of occurrences. As $\alpha$ is a sequence of itemsets, an item can occur at most once in a transaction. Hence, for a given signature item, the second type of occurrences for this item are distributed uniformly over the transactions where this item does not already occur for the first type of occurrences. Finally, the third type are the occurrences of the remaining items: the items that are not part of the signature. There is no information about these items occurrences, hence we assume them to be generated uniformly over all transactions of $\alpha$.

With these three types of occurrences, the signature model is probabilistic: all occurrences in $\alpha$ are generated according to a probability distribution that takes into account the information provided by the signature specification. Hence, we can now define the problem we are tackling:

*Problem 1.* Let $\mathbb{S}$ denote the set of signatures for all values of $k$, $\mathbb{S} = \bigcup_{k=1}^{|\alpha|} Sig(\alpha, k)$. Given a sequence $\alpha$, it follows from the MDL principle that the best signature $S \in \mathbb{S}$ is the one that minimizes the two-part encoded length of $S$ and $\alpha$, i.e.,

$$S_{MDL} = \text{argmin}_{S \in \mathbb{S}} L(\alpha, S),$$

where $L(\alpha, S)$ is the two-part encoded length that we present in the next section.

## 4 An Encoding for Signatures

As typically done in compression-based pattern mining [7], we use a two-part MDL code that leads to decomposing the total encoded length $L(\alpha, S)$ into two parts: $L(S)$ and $L(\alpha|S)$, with the relation $L(\alpha, S) = L(S) + L(\alpha|S)$. In the upcoming subsection we define $L(S)$, i.e., the encoded length of a signature, after which Subsection 4.2 introduces $L(\alpha|S)$, i.e., the length of the sequence $\alpha$ given a signature $S$. In the remainder of this paper, all logarithms are in base 2.

### 4.1 Model Encoding: $L(S)$

A signature is composed of two parts: 1) the signature items, and 2) the signature segmentation. The two parts are detailed below.

**Signature items encoding** The encoding of the signature items consists of three parts. The signature items are a subset of $\mathcal{I}$, hence we first encode the number of items in $\mathcal{I}$. A common way to encode non-negative integer numbers is to use the universal code for integers [4, 8], denoted $L_{\mathbb{N}}$[4]. This yields a code of size $L_{\mathbb{N}}(|\mathcal{I}|)$. Next, we encode the number of items in the signature, using again the universal code for integers, with length $L_{\mathbb{N}}(|\mathcal{R}|)$. Finally, we encode the items of the signature. As the order of signature items is irrelevant, we can use an $|\mathcal{R}|$-combination of $|\mathcal{I}|$ elements without replacement. This yields a length of $\log(\binom{|\mathcal{I}|}{|\mathcal{R}|})$. From $\mathcal{R}$ and $\mathcal{I}$, we can deduce $\mathcal{E}$.

**Segmentation encoding** We now present the encoding of the second part of the signature: the signature segmentation. To encode the segmentation, we encode the segment boundaries. These boundaries are indexed on the size of the sequence, hence we first need to encode the number of transactions $n$. This can be done using again the universal code for integers, which is of size $L_{\mathbb{N}}(n)$. Then, we need to encode the number of segments $|S|$, which is of length $L_{\mathbb{N}}(|S|)$. To encode the segments, we only have to encode the boundaries between two consecutive segments. As there are $|S|-1$ such boundaries, a naive encoded length would be $(|S|-1)*\log(n)$. An improved encoding takes into account the previous segments. For example, when encoding the second boundary, we know that its value will not be higher than $n-|S_1|$. Hence, we can encode it in $\log(n-|S_1|)$ instead of $\log(n)$ bits. This principle can be applied to encode all boundaries. Another way to further reduce the encoded length is to use the fact that we know that each signature segment contains at least one transaction. We can therefore subtract the number of remaining segments to encode the boundary of the segment we are encoding. This yields an encoded length of $\sum_{i=1}^{|S|-1}\log(n-(|S|-i)-\sum_{j=1}^{i-1}|S_j|)$.

*Putting everything together* The total encoded length of a signature $S$ is
$L(S) = L_{\mathbb{N}}(|\mathcal{I}|) + L_{\mathbb{N}}(|\mathcal{R}|) + \log(\binom{|\mathcal{I}|}{|\mathcal{R}|}) +$
$L_{\mathbb{N}}(n) + L_{\mathbb{N}}(|S|) + \sum_{i=1}^{|S|-1}\log(n-(|S|-i)-\sum_{j=1}^{i-1}|S_j|)$.

### 4.2 Data Encoding: $L(\alpha|S)$

We now present the encoding of the sequence given the model: $L(\alpha|S)$. This encoding relies on the refinement of the signature model into a probabilistic model presented in Section 3. To summarize, we have three separate encoding streams that encode the three different types of occurrences presented in Section 3: 1) one that encodes one occurrence of every signature item in every segment, 2) one that encodes the rest of the signature items occurrences, and 3) one that encodes the remaining items occurrences. An example illustrating the three different encoding streams is presented in Figure 2.

[4] $L_{\mathbb{N}} = \log^*(n) + \log(2.865064)$, with $\log^*(n) = \log(n) + \log(\log(n)) + \ldots$

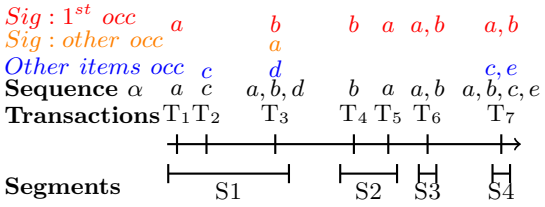


**Fig. 2.** A sequence of transactions and its encoding scheme. We have $\mathcal{R} = \{a, b\}$, $\mathcal{E} = \{c, d, e\}$ and $\mathcal{I} = \{a, b, c, d, e\}$. The first occurrence of each signature item in each segment is encoded in the red stream, the remaining signature items occurrences in the orange stream, and the items from $\mathcal{E}$ in the blue stream.

**Encoding one occurrence of each signature item in each segment** As stated in Section 3, the signature says that in each segment, there is at least one occurrence of each signature item. The size of each segment is known (from the encoding of the model, in Subsection 4.1), hence we encode one occurrence of each signature item in segment $S_i$ by encoding the index of the transaction, within segment $S_i$, that contains this occurrence. From Section 3, this occurrence is uniformly distributed over the transactions in $S_i$. As encoding an index over $|S_i|$ equiprobable possibilities costs $\log(|S_i|)$ bits and as in each segment, $|\mathcal{R}|$ occurrences are encoded this way, we encode each segment in $|\mathcal{R}| * \log(|S_i|)$ bits.

**Encoding the remaining signature items' occurrences** As presented in Figure 2, we now encode remaining signature items occurrences to guarantee a lossless encoding. Again, this encoding relies on encoding transactions where signature items occur. For each item $a$, we encode its occurrences $occ(a) = \sum_{T_i \in \alpha} \sum_{p \in T_i} \mathbf{1}_{a=p}$ by encoding to which transaction it belongs. As $S$ occurrences have already been encoded using the previous stream, there are $occ(a) - |S|$ remaining occurrences to encode. These occurrences can be in any of the $n - |S|$ remaining transactions. From Section 3, we use a uniform distribution to encode them. More precisely, the first occurrence of item $a$ can belong to any of the $n-|S|$ transactions where $a$ does not already occur. For the second occurrence of $a$, there are now only $n-|S|-1$ transactions where $a$ can occur. By applying this principle, we encode all the remaining occurrences of $a$ as $\sum_{i=0}^{occ(a)-|S|-1} \log(n-|S|-i)$. For each item, we also use $L_{\mathbb{N}}(occ(a) - |S|)$ bits to encode the number of occurrences. This yields a total length of $\sum_{a \in \mathcal{R}} L_{\mathbb{N}}(occ(a)-|S|)+\sum_{i=0}^{occ(a)-|S|-1} \log(n-|S|-i)$.

**Remaining items occurrences encoding** Finally, we encode the remaining items occurrences, i.e., the occurrences of items in $\mathcal{E}$. The encoding technique is identical to the one used to encode additional signature items occurrences, with the exception that the remaining items occurrences can initially be present in any of the $n$ transactions. This yields a total code of $\sum_{a \in \mathcal{E}} L_{\mathbb{N}}(occ(a))+\sum_{i=0}^{occ(a)} \log(n-i)$.

*Putting everything together* The total encoded length of the data given the model is given by: $L(\alpha|S) = \sum_{S_i \in S} |\mathcal{R}| * \log(|S_i|) + \sum_{a \in \mathcal{R}} L_{\mathbb{N}}(occ(a) - |S|) + \sum_{i=0}^{occ(a)-|S|-1} \log(n - |S| - i) + \sum_{a \in \mathcal{E}} L_{\mathbb{N}}(occ(a)) + \sum_{i=0}^{occ(a)} \log(n - i)$.

## 5 Algorithms

The previous section presented how a sequence is encoded, completing our problem formalization. The remaining problem is to find the signature minimizing the code length, that is, finding $S_{MDL}$ such that $S_{MDL} = \text{argmin}_{S \in \mathbb{S}} L(\alpha, S)$.

**Naive algorithm** A naive approach would be to directly mine the whole set of signatures $\mathbb{S}$ and find the signature that minimizes the code length. However, mining a signature with $k$ segments has time complexity $O(n^2 k)$. Mining the whole set of signatures requires $k$ to vary from 1 to $n$, resulting in a total complexity of $O(n^4)$. The quartic complexity does not allow us to quickly mine the complete set of possible signatures on large datasets, hence we have to rely on heuristic approaches.

To quickly search for the signature in $\mathbb{S}$ that minimizes the code length, we initially rely on a top-down greedy algorithm. We start with one segment containing the whole sequence, and then search for the segment boundary that minimizes the encoded length. Then, we recursively search for a new single segment boundary that minimizes the encoded length. We stop when no segment can be added, i.e., when the number of segments is equal to the number of transactions. During this process, we record the signature with the best encoded length. However, this algorithm can perform early segment splits that seem promising initially, but that eventually impair the search for the best signature.

### 5.1 Widening for signatures

To solve this issue, a solution is to keep the $w$ signatures with the lowest code length at each step instead of keeping only the best one. This technique is called *beam search* and has been used to tackle optimization problems in pattern mining [6]. The beam width $w$ is the number of solutions to keep at each step of the algorithm. However, the beam search technique suffers from having many of the best $w$ signatures that tend to be similar and correspond to slight variations of one signature. Here, this means that most signatures in the beam would have segmentations that are very similar. The widening technique [9] solves this issue by adding a diversity constraint into the beam. Different constraints exist [5, 6, 9], but a common solution is to add a distance constraint between each pair of elements in the beam: all pairwise distances between the signatures in the beam have to be larger than a given threshold $\theta$. As this threshold is dependent on the data and the beam width, we propose a method to automatically set its value.

Algorithm 1 presents the proposed widening algorithm. Line 3 iterates over the number of segments. Line 4 computes all signatures having $k$ segments that are considered to enter the beam. More specifically, function *Split1Segment* computes the direct refinements of each of all signatures in $BestKSign$. A direct refinement of a signature corresponds to splitting one segment in the segmentation associated with that signature. Line 5 selects the refinement having the smallest code length. If several refinements yield the smallest code length, one of these refinements is chosen at random. Lines 8 to 11 perform the widening step by adding new signatures to the beam while respecting the pairwise distance constraint. Line

**Algorithm 1** Widening algorithm for signature code length minimization.

```
1: function SIGNATURE MINING(α = ⟨T_1, ..., T_n⟩, β, w)
2:     BestKSign = ∅, BestSign = ∅
3:     for k = 1 → n do
4:         AllKSign = Split1Segment(BestKSign)
5:         S_opt = argmin_{S∈AllKSign} L(α, S)
6:         BestSign = BestSign ⋃{S_opt}
7:         BestKSign = {S_opt}
8:         θ = threshold(β, w,AllKSign)
9:         while S_opt ≠ ∅ and |BestKSign| < w do
10:            S_opt = argmin_{S∈AllKSign} L(α, S), ∄S_i ∈ BestKSign, d(S_i, S) ≤ θ
11:            BestKSign = BestKSign ⋃{S_opt}
12:    return argmin_{S∈BestSign} L(α, S)
```

**Algorithm 2** Distance threshold computation.

```
1: function THRESHOLD(β, w, AllSign)
2:     KBest = β * |AllSign|
3:     BestS = GetBestSign(AllSign, KBest)
4:     return argmin_θ{N(θ), N(θ) = |{S ∈ BestS, d(S, BestS[0]) < θ}|, N(θ) ≥ |BestS|/w}
```

8 computes the distance threshold ($\theta$) depending on the diversity parameter ($\beta$), the beam width ($w$), and the current refinements. Algorithm 2 presents the details of the threshold computation. With this threshold, we recursively add a new element in the beam, until either the beam is full or no new element can be added (line 9). Lines 10 and 11 add the signature having the smallest code length and being at a distance of at least $\theta$ to any current element of the beam. Line 12 returns the best overall signature we have encountered.

**Distance between signatures** We now define the distance measure for signatures (used in line 10 of Algorithm 1). As the purpose of the signature distance is to ensure diversity in the beam, we will use the segmentation to define the distance between two elements of the beam, i.e., between two signatures. Terzi et al. [10] presented several distance measures for segmentations. The *disagreement distance* is particularly appealing for our purposes as it compares how transactions belonging to the same segment in one segmentation are allocated to the other segmentation. Let $S_a = \langle S_{a1} \dots S_{ak} \rangle$ and $S_b = \langle S_{b1} \dots S_{bk} \rangle$ be two $k$-segmentations of a sequence $\alpha$. We denote by $d(S_a, S_b)$ the disagreement distance between segmentation $a$ and segmentation $b$. The disagreement distance corresponds to the number of transaction pairs that belong to the same segment in one segmentation, but that are not in the same segment in the other segmentation. Techniques on how to efficiently compute this distance are presented in [10].

**Defining a distance threshold** Algorithm 1 uses a distance threshold $\theta$ between two signatures, that controls the diversity constraint in the beam. If $\theta$ is equal to 0, there is no diversity constraint, as any distance between two different

signatures is greater than 0. Higher values of $\theta$ enforce more diversity in the beam: good signatures will not be included in the beam if they are too close to signatures already in the beam. However, setting the $\theta$ threshold is not easy. For example $\theta$ depends on the beam width $w$. Indeed, with large beam widths, $\theta$ should be low enough to allow many good signatures to enter the beam.

To this end, we introduce a method that automatically sets the $\theta$ parameter, depending on the beam width and on a new parameter $\beta$ that is easier to interpret. The $\beta$ parameter ranges from 0 to 1 and controls the strength of the diversity constraint. The intuition behind $\beta$ is that its value will approximately correspond to the relative rank of the worst signature in the beam. For example, if $\beta$ is set to 0.2, it means that signatures in the beam are in the top-20% in ascending order of code length. Algorithm 2 details how $\theta$ is derived from $\beta$ and $w$; this algorithm is called by the *threshold* function in line 8 of Algorithm 1.

Knowing the set of all candidate signatures that are considered to enter the beam, we retain only the proportion $\beta$ of the best signatures (line 3 of Algorithm 2). Then, in line 4 we extract the best signature. Finally, we look for the distance threshold $\theta$ such that the number of signatures within a distance of $\theta$ from the best signature is equal to the number of considered signatures divided by the beam width $w$ (line 5). The rationale behind this threshold is that since we are adding $w$ signatures to the beam and we want to use the proportion $\beta$ of the best signatures, the distance threshold should approximately discard $1/w$ of the proportion $\beta$ of the best signatures around each signature of the beam.

## 6 Experiments

This section, analyzes runtimes and code lengths of variants of our algorithm on a real retail dataset[5]. We show that our method runs significantly faster than the naive baseline, and give advice on how to choose the $w$ and $\beta$ parameters. Next, we illustrate the usefulness of the encoding to analyze retail customers.

### 6.1 Algorithm runtime and code length analysis

We here analyze the runtimes and code lengths obtained by variants of Algorithm 1. 3000 customers having more than 40 baskets in the Instacart 2017 dataset are randomly selected[6]. Customers having few purchases are less relevant, as we are looking for purchase regularities. These 3000 customers are analyzed individually, hence the algorithm is evaluated on different sequences.

**Code length analysis** To assess the performance of the different algorithms, we analyze the code length yielded by each algorithm on each of these 3000 customers. We evaluate different instances of the widening algorithm with different beam widths $w$ and diversity constraints $\beta$. The resulting relative mean code

[5] Code is available at `https://bitbucket.org/clement_gautrais/mdl_signature_ida2020/`

[6] The Instacart Online Grocery Shopping Dataset 2017, Accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on 05/04/2018

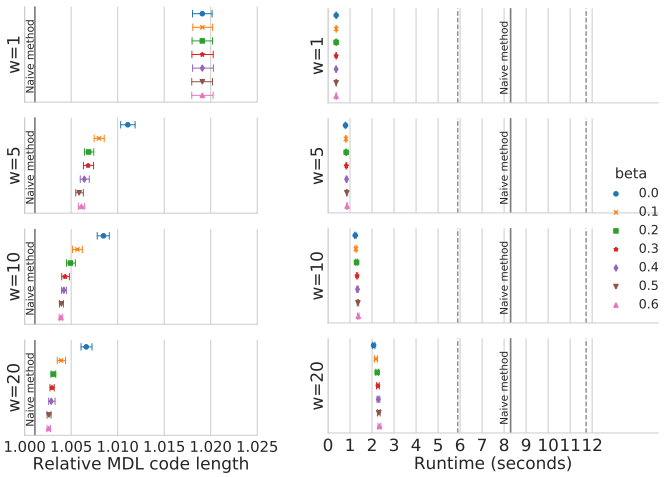


**Fig. 3. Left**: Mean relative code length for different instances of the widening algorithm. For each customer, the relative code length is computed with regard to the smallest code length found for this customer. Averaging these lengths across all customers gives the mean relative code length. The $\beta$ parameter sets the diversity constraint and $w$ the beam width. The solid black line shows the mean code length of the naive algorithm. Bootstrapped 95% confidence intervals [1] are displayed. **Right**: Mean runtime in seconds for different instances of the widening algorithm. The dotted black lines shows a bootstrapped 95% confidence interval of the naive algorithm's mean runtime.

lengths per algorithm instance are presented in Figure 3 left. When increasing the beam width, the code length always decreases for a fixed $\beta$ value. This is expected, as increasing the beam size allows the widening algorithm to explore more solutions. As increasing the beam size improves the search, we recommend setting it as high as your computational budget allows you to do.

Increasing the $\beta$ parameter usually leads to better code lengths. However, for $w = 5$, higher $\beta$ values give slightly worse results. Indeed, if $\beta$ is too high, good signatures might not be included in the beam, if they are too close to existing solutions. Therefore, we recommend setting the $\beta$ value to a moderate value, for example between 0.3 and 0.5. A strong point of our method is that it is not too sensitive to different $\beta$ values. Hence, setting this parameter to its optimal value is not critical. The enforced diversity is highly relevant, as a fixed beam size with some diversity finds code lengths that are similar to the ones found by a larger beam size with no diversity. For example, with $w = 5$ and $\beta = 0.3$, the code lengths are better than with $w = 10$ and $\beta = 0$. As using a beam size of 5 with $\beta = 0.3$ is faster than using a beam size of 10 with $\beta = 0$, it shows that using diversity is highly suited to decrease runtime while yielding smaller code lengths.

**Runtime analysis** We now present runtimes of different widening instances in Figure 3 right. The beam width mostly influences the runtime, whereas the $\beta$ value has a smaller influence. Overall, increasing $\beta$ slightly increases computation

time, while yielding a noticeable improvement in the resulting code length, especially for small beam sizes. Our method also runs 5 to 10 times faster than the naive method. In this experiment, customers have a limited number of baskets (at most 100), thus the $O(n^4)$ complexity of the naive approach exhibits reasonable runtimes. However in settings with more transactions (retail data over a longer period for example), the naive approach will require hours to run, and the performance gain of our widening approach will be a necessity. Another important thing is that the naive method has a high variability in runtimes. Confidence intervals are narrow for the widening algorithm (they are barely noticeable on the plot), whereas it spans over 5 seconds for the naive algorithm.

### 6.2 Qualitative analysis

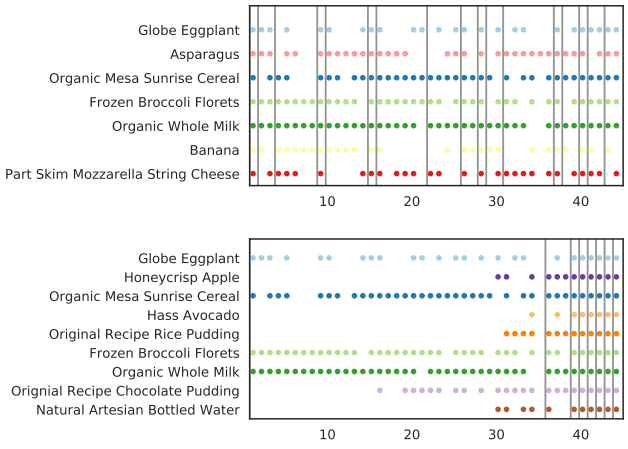


**Fig. 4.** Example of two signatures found by our algorithms. Gray vertical lines are segments boundaries and each dot represents an item occurrence in a purchase sequence. **Top**: best signature (code length of 5221.33 bits) found by the widening algorithm, with $w = 20$ and $\beta = 0.5$. **Bottom**: signature found by the beam search algorithm: $w = 1$ and $\beta = 0$, with a code length of 5338.46 bits (the worst code length).

Figure 4 presents two signatures of a customer, to illustrate that signatures are of practical use to analyze retail customers, and that finding signatures with smaller code lengths is of interest. We use the widening algorithm to get a variety of good signatures according to our MDL encoding. The top signature in Figure 4 is the best signature found: it has the smallest code length. This signature seems to correctly capture the regular behavior of this customer, as it contains 7 products that are regularly bought throughout the whole purchase sequence. Knowing these 7 favorite products, a retailer could target its offers. The segments also give some information regarding the temporal behavior of this customer. For

example, because segments tend to be smaller and more frequent towards the end of the sequence, one could guess that this customer is becoming a regular.

On the other hand, the bottom signature is significantly worse than the top one. It is clear that it mostly contains products that are bought only at the end of the purchase sequence of this customer. This phenomenon occurs because the beam search algorithm, with $w = 1$, only picks the best solution at each step of the algorithm. Hence, it can quickly get stuck in a local minimum. This example shows that considering larger beams and adding diversity is an effective approach to optimize code length. Indeed, having a large and diverse beam is necessary to have the algorithm explore different segmentations, yielding better signatures.

## 7 Conclusions

We tackled the problem of automatically finding the best number of segments for signature patterns. To this end, we defined a model selection problem for signatures based on the minimum description length principle. Then, we introduced a novel algorithm that is an instance of widening. We evaluated the relevance and effectiveness of both the problem formalization and the algorithm on a retail dataset. We have shown that the widening-based algorithm outperforms the beam search approach as well as a naive baseline. Finally, we illustrated the practical usefulness of the signature on a retail use case. As part of future work, we would like to study our optimization techniques on larger databases (thousands of transactions), like online news feeds. We would also like to work on model selection for *sets* of interesting signatures, to highlight diverse recurrences.

## References

1. Davison, A.C., Hinkley, D.V., et al.: Bootstrap methods and their application, vol. 1. Cambridge university press (1997)
2. Gautrais, C., Cellier, P., Quiniou, R., Termier, A.: Topic signatures in political campaign speeches. In: Proc. of EMNLP 2017. pp. 2342–2347 (2017)
3. Gautrais, C., Quiniou, R., Cellier, P., Guyet, T., Termier, A.: Purchase signatures of retail customers. In: PAKDD. pp. 110–121. Springer (2017)
4. Grünwald, P.D.: The minimum description length principle. MIT press (2007)
5. Ivanova, V.N., Berthold, M.R.: Diversity-driven widening. In: International Symposium on Intelligent Data Analysis. pp. 223–236. Springer (2013)
6. van Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Mining and Knowledge Discovery **25**(2), 208–242 (2012)
7. van Leeuwen, M., Vreeken, J.: Mining and using sets of patterns through compression. In: Frequent Pattern Mining, pp. 165–198. Springer (2014)
8. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of statistics pp. 416–431 (1983)
9. Shell, P., Rubio, J.A.H., Barro, G.Q.: Improving search through diversity. In: Proc. of the AAAI Nat. Conf. on Artificial Intelligence. pp. 1323–1328. AAAI Press (1994)
10. Terzi, E.: Problems and algorithms for sequence segmentations. Ph.D. thesis (2006)
11. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery **23**(1), 169–214 (2011)